§0 Introduction & Motivation

- "Virtues" of Computer Science and Mathematics
- Data abstraction in Computer Science and Math
- Hardware data types / mathemat. structures
- Programming vs. Software Engineering
- IEEE 754 / Classical Numerics
- Reliability in Numerics, Examples
- Numerical Folklore and Myths

Theoretical Computer Science *and* Mathematics

KAIST

"***Virtues***":

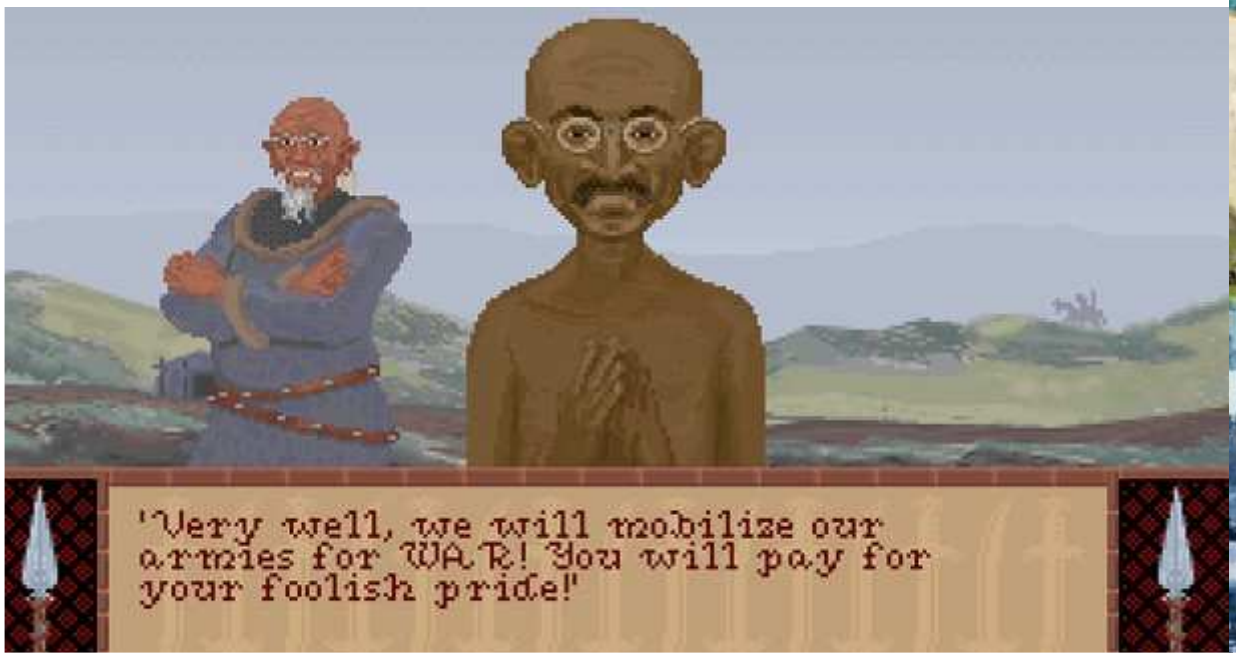
- problem specification
- formal semantics
- algorithm design
- and analysis (correctness, efficiency)
- optimality proof

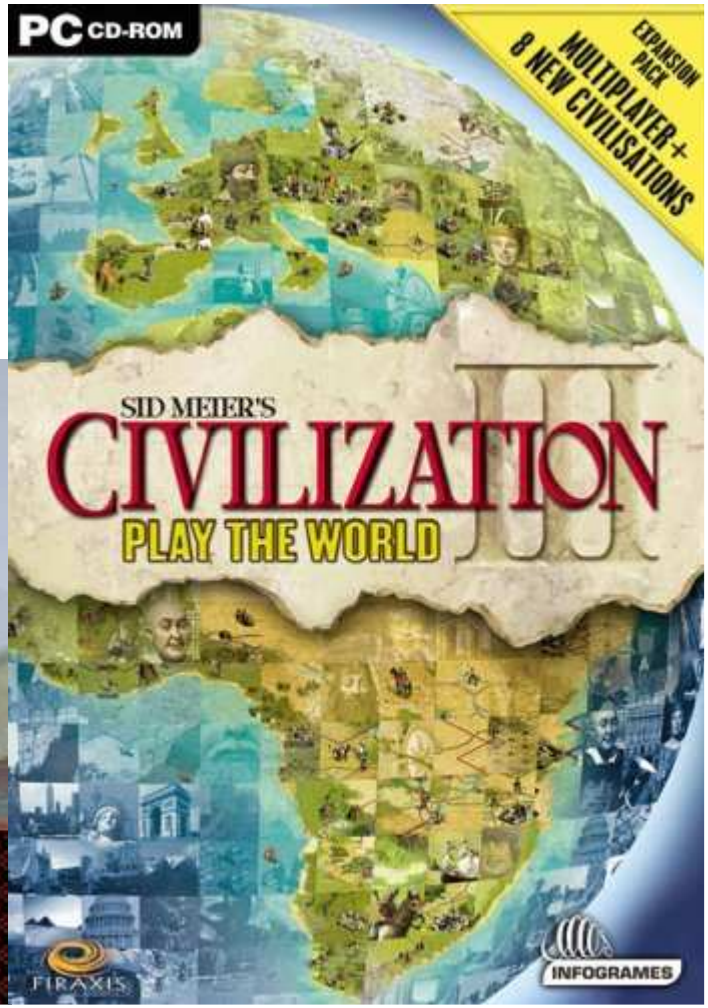
Computer Game *Civilization*

In the simulation, each country (leader) is described by quantitative attributes that govern its "AI" behavior.

Initially Gandhi's **`aggression`** :=1

When a country adopts democracy, its **`aggression`** *de*creases by 2.





Data Abstraction in CS and Math

KAIST

hi-level PL

Interpreter

Compiler

Library

Christofides' Algorithm

Prim's Algorithm

Fibonacci Heap

linked data structure

totally ordered keys

continuous/ smooth/ integrable *functions*

$\mathbb{G}$

$\mathbb{P}$

$\mathbb{C}$

`double` ≠ $\mathbb{R}$

$\mathbb{A}$

$\mathbb{Q}$

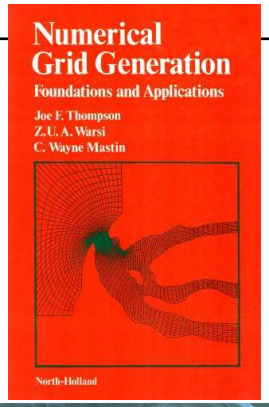
`byte` ≠ $\mathbb{Z}$

$\mathbb{N}$

Hardware vs. Math. Data Types

$\mathbb{Q}$, as opposed to $\mathbb{R}$, violates:

- intermed. value theorem
- compactness of [0,1]
- fixed point theorems,
- logical completeness,
- **assoc./distrib.laws**



continuous/
smooth/
integrable
functions

$\mathbb{G}$

$\mathbb{P}$

$\mathbb{C}$

`double` $\neq \mathbb{R}$

long multiplication
Karatsuba
Toom, Cook
Strassen-Schönhage

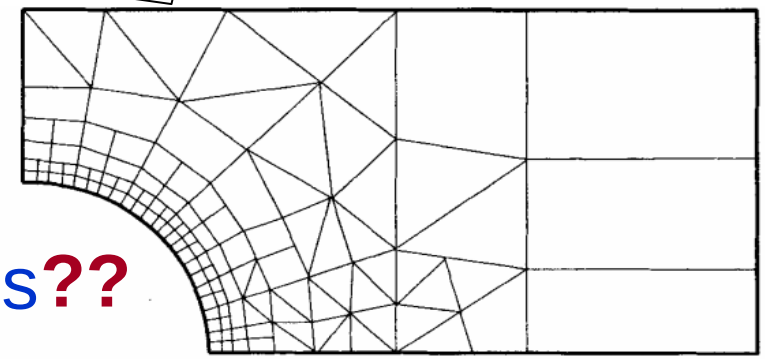
$\mathbb{A}$

$\mathbb{Q}$

Don't test for equality!

How about *in*equality "**<**" ?

$x=0 \Leftrightarrow \neg(x<0) \wedge \neg(x>0)$



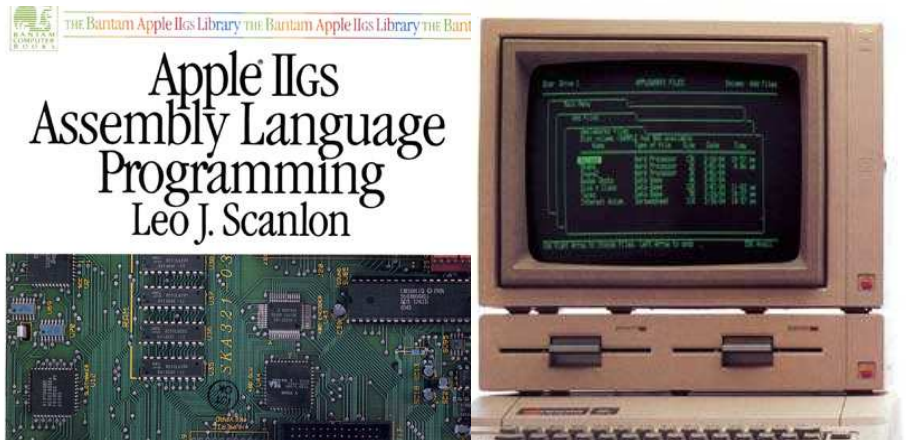
Semantics of function/higher types**??**

Programming vs. Software Engineering

- *empirical* correctness
- *hardware* data type **`byte`**
- *in*convenient/ *un*elegant
- *absolute* performance
- *low*-level prog.language

$\neq \mathbb{Z}$

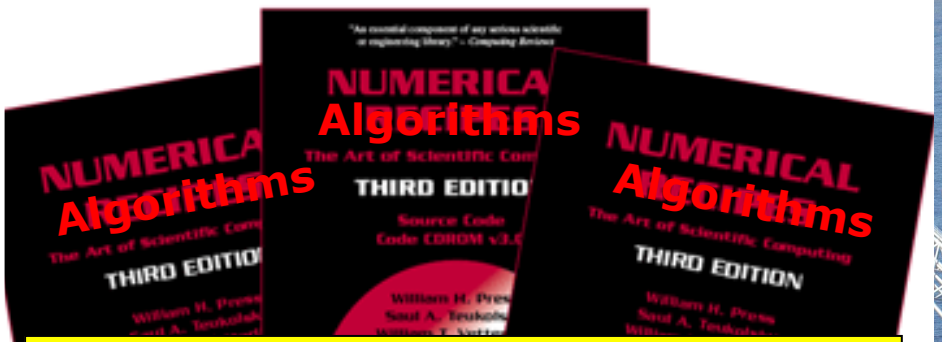
→bugs



1. Rigorous specification
2. Algorithm design&analysis (*always* correct & efficient)
3. Proof of optimality (P/NP)
4. Design/build on axiomatic ***Abstract Data Types***
5. in obj.-oriented high-level prog.language (semantics)
6. Formal verification
7. Implementation/coding

IEEE 754 / Classical *Numerics*

- *empirical* correctness
- *hardware* data type **`float/double`** ≠ℝ
- *in*convenient/ *un*elegant →bugs



„`nag_opt_one_var_deriv`“
normally computes a sequence of x values which *tend in the limit* to a minimum of F x

7. Implementation/coding

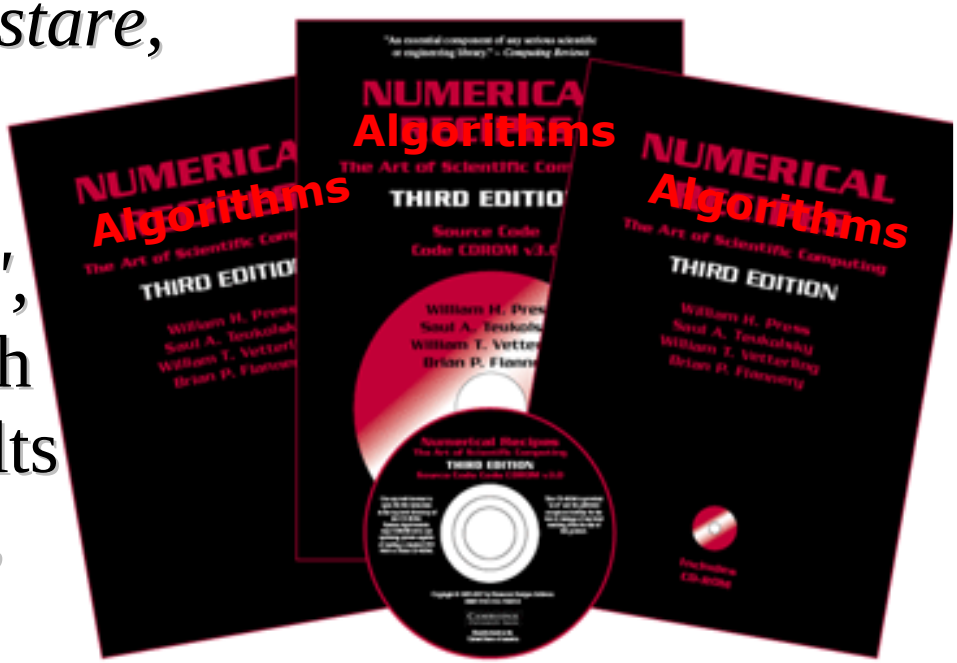
Reliability in Numerical Software?

KAIST

Peter Linz (Courant Institute), p.412, Bull. AMS vol.19:2

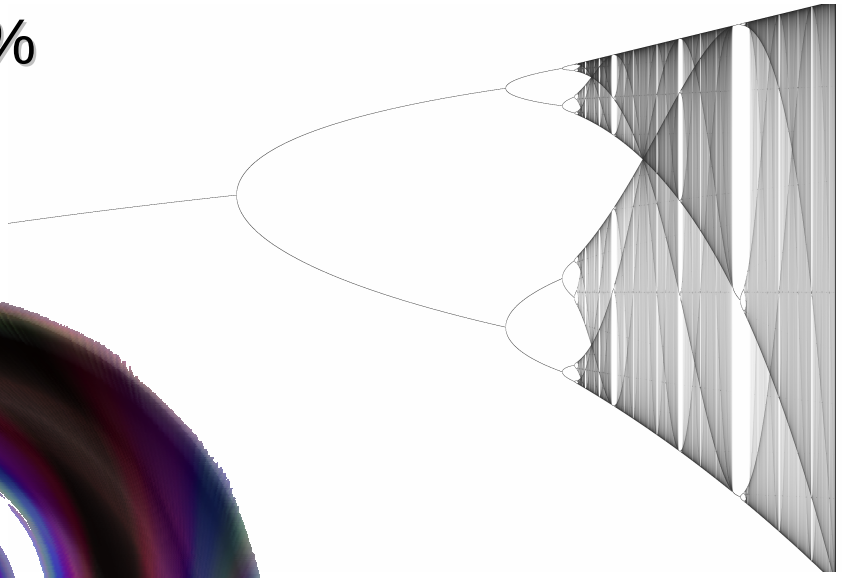
«*Over the years, I have sat on many Ph.D. qualifying examinations or dissertation defenses for engineering students whose work involved a significant amount of numerical computing. In one form or another, I invariably ask […]:* "How do you know that your answers are as accurate as you claim?" *[...]*

After an initial blank or hostile stare, I usually get an answer like
"I tested the method with some simple examples and it worked",
"I repeated the computation with several values of *n* and the results agreed to three decimal places",
or more lamely, "the answers looked like what I expected".



Example Prog. Codes

$x_n \to x_{n+1} := r \cdot x_n \cdot (1\text{-}x_n)$ error ≤10%



float: $n \leq 30$, double: $n \leq 85$,
long double: $n \leq 200$
rational: ***n≤24***

```
MATLAB function
  y=jmmuller(m)
x = vpa(11/2);
y = vpa(61/11);
while m>1;
 t = vpa(111 -
   (1130-3000/x)/y);
 x=y; y=t; m=m-1;
 end; end
```

→ **100**

$x_0 := 11/2, \quad x_1 := 61/11$

$x_{m+1} := 111 - (1130\text{-}3000/x_{m-1})/x_m \to \mathbf{6}$

Numerical Folklore & Myths

Must not test for equality "=="

How about inequality "<" ?

$x=0 \Leftrightarrow \neg(x<0) \wedge \neg(x>0)$

$\rightarrow$ *multivalued* semantics

[Specker'59] There is a computable C^∞ $f:[0;1]\rightarrow[0;1]$ attaining its minimum in *no* computable point

[Orevkov'63]: There is a computable continuous $f:[0;1]^2\rightarrow[0;1]^2$ with *no* computable fixed point.

[Pour-El&Richards'89]
There is a computable initial condition f s.t. solution $u(1)$ is *not* computable
(contains encoding of Halting problem)

$$\Delta u = \ddot{u}$$
$$u(0) = f$$
$$u'(0)=0$$

Background Self-Check

- Convergent sequence
- Continuous function
- Compact subset
- Metric space
- Logic?

- C++
- Unix/Linux

- Halting Problem
- Models of Computation
- Oracle machine
- $\mathcal{NP}$, reduction
- Approx. algorithm

§0 Introduction & Motivation

- "Virtues" of Computer Science and Mathematics
- Data abstraction in Computer Science and Math
- Hardware data types / mathemat. structures
- Programming vs. Software Engineering
- IEEE 754 / Classical Numerics
- Reliability in Numerics, Examples
- Numerical Folklore and Myths